Latency and Bandwidth Utilization Analysis of the FlexRay Static Segment

Jasmu Jacob #1, Dr. Shelbi Joseph #2

#1 M.Tech Student, Information Technology, School of Engineering, CUSAT, Kerala, India
#2 Associate Professor, Information Technology, School of Engineering, CUSAT, Kerala, India

Abstract – Autonomous vehicular system, which is a multi- criticality cyber-physical-system incorporates numerous Electronic Control Units (ECUs) and hence require the use of a unified cross-domain architectural design. This has contributed to the introduction of Automotive Open Systems Architecture (AUTOSAR) which facilitates the inter-communication of these units regardless of the technology used. FlexRay communication protocol, which comes under AUTOSAR offers high reliability compared to TTCAN, ByteFlight etc due to its flexibility and deterministic nature. This paper demonstrates a survey on different verification methodologies done on time-triggered architecture and analyses the slot performance of the FlexRay protocol concerning latency and bandwidth utilization using a modeling and simulation tool -VisualSim.


I. INTRODUCTION

The present-time automobile industry is rapidly advancing up and becoming more and more complex and hence it must satisfy uncomprising safety standards [1]. Furthermore, the introduction of autonomous/driverless vehicles has resulted in a hike in the amount of Electronic Control Units (ECUs) used in each vehicle. For instance, the control system of a car consists of almost hundred different computational units and they could be structured as different distributed application subsystem (DAS) such as the breaking system, the engine-control systems, the weather-control systems etc. These different systems (DASs) are manufactured by different companies, which consists of its own standalone hardware base consisting of Electronic Control Units (ECUs) or nodes and are interconnected by physical communication channels [2]. As the number of control units increases, lot of hardware are required to integrate these systems, thereby increasing the failure rates. In order to satisfy the requirements of the increased number of ECUs, higher bandwidth is required for the channel. Moreover, there arises the need for communication protocols as a mechanism for the centralized control of data transmission between these units [3]. Also the time delay of processing the data between the ECUs must be reduced. The protocols which use the time-triggered architecture perform a significant role, especially in safety-critical and real-time applications. Among these, the FlexRay protocol is more flexible and deterministic and integrates both static and dynamic message transmission and thus combines the added benefit of synchronous as well as asynchronous protocols [3]. FlexRay consists of dual channels each having bandwidth of 10Mbps [4]. Another critical issue in such an integrated architecture is fault isolation and error propagation. To handle this, FlexRay has a deterministic fault-tolerance mechanism.

In order to minimize the issues and to efficiently run the system, the protocols that are used by safety-critical applications need to be thoroughly verified. For a larger system such as a time-triggered-architecture-based one, we can verify its algorithms using formal verification methods [5] and it has been seen that this technique is effective on verifying each components of such systems especially the critical application modules. Verifying a communication protocol emphasizes its credibility and correctness before its implementation. There are different protocol-verifying techniques from which we can choose the most suited one according to the program or finite state machine specification that is used during the modeling of these protocols. The proposed system analyzes individual FlexRay channel, nodes, and slot performance with respect to latency and throughput using VisualSim Architect.

The remaining part of this paper is structured as follows. Section II describes the time-triggered architecture in detail and makes a comparison of different protocols. Section III provides the background and related works of this paper. Section IV discusses about the FlexRay protocol. Section V presents the experimental evaluation. In Section VI, the results obtained from the simulation and analysis is summarized. Finally, Section VII concludes this paper.

II. TIME-TRIGGERED ARCHITECTURE

The time-triggered architecture [21] has been evolved as a significant part of many distributed computing systems mainly for safety-critical and real-time applications such as autonomous vehicular control systems. TTA, FlexRay, and TTCAN are a few examples of such time-triggered architectures that have been popular now-a-days. The development of time-triggered architecture was started in 1979 at TU, Berlin, and continued at TU Wien since 1982. The architecture is brought to industry by TT Tech, Vienna in

IJSRCAMS

Volume 7, Issue 5 (September 2018)
1998. TTA is presently deployed in the application domain such as aerospace (Honeywell, Boeing, Airbus), railway signaling in Thales, automotive drive-by-wire prototypes (Audi, BMW) etc. The time-triggered architecture was also the first step towards the European cross-domain embedded system architecture GENESYS. The time-triggered architecture is a deterministic one with separation of subsystems which enables us to build complex systems out of simple components. Here large applications are decomposed into several independent systems in clusters thereby making it in a simplified form. The time-triggered architecture has a highly deterministic real time behavior which ensures the users’ physical safety by meeting the real-time requirements.

A. Structure of a Time-triggered Architecture

The primary building unit of this architecture is usually called a node. A node consists of a processor having a communication controller, memory unit associated with it, an operating system, input-output unit and the application software. The architecture consists of many such nodes synchronizing with each other and exchanging messages among them. TTA uses either bus topology or star topology[25]. The medium of communication, whether it may be bus or star, is shared among the nodes. Based on a task schedule, which is determined earlier, the time-triggered architecture performs a set of tasks. This is made possible by using an interrupt which is linked to the periodic overflow of a timer. This architecture uses a lot of a priori timing information for different services such as fault detection, error correction, message identification etc. The time-triggered architectures uses a variety of algorithms such as start-up algorithm, clock synchronization, window timing, non-blocking write algorithm, group membership protocol, clique avoidance etc [21].

B. Features of a Time-triggered Architecture

This framework possesses the property of composability[23] which is the systematic construction of a system out of its subsystems or components at a higher abstraction level thereby independent functions can be integrated with a minimal effort. Also the components can be replicated effectively which enables better fault tolerant mechanism. The time-triggered architecture mainly focuses on providing distributed computing platform, unification of interfaces, composability, scalability, fault handling etc. The TTA is highly scalable in the sense that a particular function of the system component does not depend on the size of the system. It provides a fault-tolerant communication between the nodes of each cluster by the use of TDMA [1].

C. Advantages of a Time-triggered Architecture

Time-triggered architecture aims at integrating components and distributing their functionalities through the network, so the number of nodes (ECUs) in the system can be reduced. Because in an integrated architecture, it is needless to place sensors on each individual components, thereby, reducing its number and cost. This architecture thus provides consistency of states; and the system facilitates enormous stability and reliability [24]. It has also been proved that this architecture is highly safe especially to meet the requirements of safety-critical applications. One of the advantages of time-triggered architecture over an event-triggered architecture is that in an event-triggered asynchronous architecture, it is difficult to distinguish a slow node from a failed node, which makes the solution to the membership problem difficult. A comparison of different protocols is shown in Table I.

<table>
<thead>
<tr>
<th>Protocols</th>
<th>Bandwidth</th>
<th>Scheduling</th>
<th>Response</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIN</td>
<td>10 Kbits/s</td>
<td>Static</td>
<td>Soft real time</td>
<td>Low</td>
</tr>
<tr>
<td>CAN</td>
<td>125 Kbits/s</td>
<td>Event-Triggered only</td>
<td>Soft real time</td>
<td>Medium</td>
</tr>
<tr>
<td>TTCAN</td>
<td>10 Mbits/s</td>
<td>Event and Time-triggered</td>
<td>Hard real time</td>
<td>High</td>
</tr>
<tr>
<td>FlexRay</td>
<td>10 Mbits/s</td>
<td>Event and Time-triggered</td>
<td>Hard real time</td>
<td>High</td>
</tr>
</tbody>
</table>

III. RELATED WORKS

This section mentions some of the previous works done for the verification purpose of the time-triggered architecture and FlexRay protocol.

Mejdi et al. [6] designed a FlexRay Controller in which the SDL diagrams are translated into StateFlow diagrams. Here, ModelSim was used to verify the controller blocks. Procedures for the Macrotick block Generation and its verification by ModelSim has been also given in this paper. Armengaud et al. [8] focus on the review of the nature of time-triggered architecture within an automotive software system, and also on the identification of integration, validation and deployment challenges of a communication protocol, FlexRay. Finally, a resource-optimized FlexRay driver was developed. Chen et al. [9] presented a solution to overcome the issue of inefficiency of existing TTA-based protocols while retaining the level of fault tolerance and reliability. A new protocol called INCUS is introduced here, which is a real-time safety-critical system protocol based on TDMA schedule. INCUS significantly improves bandwidth utilisation over the traditional schemes and simultaneously reduces the gross overhead time by almost 90%.

Ammar et al. [10] has modeled and verified real-time network framework using PRISM model checker. This work can be utilized as a reference to further improve the reliability on the TTE protocol. This approach easily identifies the faulty
states or week points within the network that might cause extensive damage to a real-life application. Saha et al. [11] focused on various efforts for verifying the start-up algorithms formally. This was useful to verify small but critical components. Furthermore, this work emphasized that the verification of the whole system is difficult. Froeschauer et al. [12] investigated the FlexRay communication protocol for automation and control. FlexRay is also compared to other field bus systems and figured out some features which are currently not supported by available field bus systems. Muhammed et al. [13] explored a star topology, which is also fault-tolerant and modified shared clock algorithm for CAN-based protocol. The star topology prevents faults travelling to the remaining star network. However, error handling capabilities are not introduced in this work. Keating et al. [14] have applied a SPIN model checker to analyze the TTCAN protocol and has uncovered and corrected a number of potentially costly issues. Ran et al. [15] have verified TTCAN protocol by proposing a Timed Communicating Sequential Process (CSP) that captures the exception in the protocol. Sagstetter et al. [16] addressed the problem of scheduling the static segment of the FlexRay protocol. This work integrates the scheduling features existing FlexRay 2.1 into FlexRay 3.0.

Chang et al. [17] presented an efficient software component approach to validate the FlexRay protocol. This paper also establishes test architecture to verify the timing constraint and calculate the response time of messages. Bergenhem [18] presented a status protocol which monitors the operations in a time-triggered system. This protocol is checked using SPIN model checker with N nodes and P processes and shown that the protocol can handle single as well as double failures during a cycle. Kang et al. [19] verified a group membership algorithm for the FlexRay protocol using UPPAAL model checker. The software development follows model-based techniques using modeling environments like EAST-ADL and Stateflow. These environments provide extensive facilities for model simulation, using which models are validated by simulating them over typical scenarios and input sequences. Application level simulation and system level testing cannot provide absolute correctness guarantee. Zhan-Yao Gu et al. [20] conducted a timing analysis test and adopted a slot multiplexing mechanism to increase bandwidth utilization. This work was very helpful in calculating the worst-case response time of message.

IV. FLEXRAY PROTOCOL

FlexRay [24] is an intra-vehicular communication protocol which took arose from the need of a high-speed bus for vehicular applications. FlexRay is intended to light up the pivotal automotive demands like availability, dependability, flexibility, and speed to enhance the leading intra-vehicular networking standards like CAN, LIN, and MOST. FlexRay uses a combination of event- and time-triggered architecture and supports dual channel bus system with a bandwidth up to 10 Mbps in each channel. It is mainly suitable for distributed, safety-critical and fault tolerant systems. FlexRay follows the AUTOSAR specifications which maintain modular structure and interfaces thus making the integration of components designed on different software platforms easier.

A. The FlexRay Stack

The FlexRay stack [26] consists of a bus driver, Network Manager, interface, and the FlexRay Transport Protocol. The FlexRay protocol stack diagram is shown in Fig.1. If a data packet cannot be sent as one Protocol Data Unit (PDU), then the FlexRay protocol segments the data and reassembles them at the receiver side. FlexRay works on star and bus topologies. For critical systems, redundant/multiple star topology helps in mitigating single-node failures. It facilitates both static and dynamic scheduling [27]. Static segment helps in maintaining the timing constraints while dynamic segment provides event-triggered message scheduling.

B. Data communication through FlexRay Controller

Fig 1. Block diagram of a FlexRay Protocol Stack

Fig 2. Data communication through a FlexRay Controller
The block diagram showing the data communication through a FlexRay controller is shown in Fig.2. Data from the application module [30] is sent through AUTOSAR COM to the Protocol Data Unit Router (PduR). The PDU Router then transfers the data to FlexRay Transport Protocol Unit (FrTp), which then delivers the data to the Electronic Control Unit. Then the data is finally conveyed to the application through the PDU router.

V. EXPERIMENTAL EVALUATION

The proposed work concentrates on the analysis of latency and bandwidth utilization of the static segment of the FlexRay protocol. For that VisualSim is used as the simulation tool, which can be effectively utilized for the performance analysis of FlexRay Protocol. An automotive library is provided by this tool that could be used to validate the architecture and optimize the slot assignments of FlexRay protocol.

A. Simulation and Analysis

In order to analyze the performance of the system, all dual ports are configured in accordance to the bus-slot configuration. The communication controller defines the slot configuration, whether it may be static or dynamic, for all the dual ports in the system. The bus statistics for each bus are generated for the analysis. These include the percentage frame utilization of both channels, aggregate throughput in kilo bits per second (Kbps) and latency in seconds (s) for each channel and each slot. There are several modeling parameters for VisualSim that can be continuously altered in order to meet performance optimization such as Transmission Start Sequence, Frame Start Sequence, Byte Start Sequence, Frame Ending Sequence, Dynamic Trailng Sequence etc. The bus operation depends on the data from the frame encoding for both segments – static and dynamic. The maximum number of static slots parameter is set as 1023 in the FlexRay specification and the maximum number of bytes per static slot parameter is 254. The parameters used are summarized in Table II.

B. Equations

The various equations used for the calculations are listed below:
1. One bit time in sec: 1.0 / (FlexRay_Speed in Mbps * 1.0E06)
2. Static_Data_Intvl_Time: (1.0E-06 / FlexRay_Speed_Mbps) * 7000.0
3. Dynamic_Data_Max_Rndm_Intvl_Time: Static_Data_Intvl_Time + (1.0E-06 / FlexRay_Speed_Mbps) * 100.0
4. Static Offset = (TransmissionStartSequence + FrameStartSequence + ByteStartSequence) * Bit Time
5. Dynamic Offset = (FrameEndSequence + TransmissionStartSequence + FrameStartSequence + ByteStartSequence) * Bit Time

The input data for the evaluation consists of data generated from the sensors both from the static and dynamic source.

Data from the static source:
- Start_time: 1.0E-06
- Mean_time: Static_Data_Intvl_Time
- Spread_time: 1.0E-05
- Random_seed: 123457L

Data from the dynamic source:
- Start_time: rand(Static_Data_Intvl_Time, Dynamic_Data_Max_Rndm_Intvl_Time)
- Mean_time: Static_Data_Intvl_Time
- Spread_time: Dynamic_Data_Max_Rndm_Intvl_Time
- Random_seed: (Node_ID_Start + 0L*123457L)
- Time_Distribution: Uniform(Mean, Spread)

TABLE II. THE PARAMETERS OF THE DESIGNED FLEXRAY CONTROLLER

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>FlexRay_Bus_Name</td>
<td>“Flex_01”</td>
</tr>
<tr>
<td>FlexRay_Speed_Mbps</td>
<td>10.0</td>
</tr>
<tr>
<td>Number_Static_Slots</td>
<td>16</td>
</tr>
<tr>
<td>Bytes_per_Static_Slot</td>
<td>16</td>
</tr>
<tr>
<td>Number_Dynamic_Slots</td>
<td>17</td>
</tr>
<tr>
<td>Bytes_per_Dynamic_Slot</td>
<td>Bytes_per_Static_Slot*1</td>
</tr>
<tr>
<td>Transmission_Start_Sequence</td>
<td>8 Bit Times</td>
</tr>
<tr>
<td>Frame_Start_Sequence</td>
<td>1 Bit Time</td>
</tr>
<tr>
<td>Byte_Start_Sequence</td>
<td>2 Bit Times</td>
</tr>
<tr>
<td>Dynamic_Trailing_Sequence</td>
<td>4 Bit Times</td>
</tr>
<tr>
<td>Channel_1_Offset</td>
<td>0</td>
</tr>
<tr>
<td>Channel_2_Offset</td>
<td>1</td>
</tr>
<tr>
<td>Channel_1_ON</td>
<td>True</td>
</tr>
<tr>
<td>Channel_2_ON</td>
<td>True</td>
</tr>
<tr>
<td>Idle_Time</td>
<td>0 Bit Time</td>
</tr>
<tr>
<td>Max_Port_Queue_Length</td>
<td>1</td>
</tr>
<tr>
<td>FlexRay_Statistics_Samples</td>
<td>1.0</td>
</tr>
<tr>
<td>DS_Field_Name_for_Bytes</td>
<td>“Payload_Bytes”</td>
</tr>
<tr>
<td>Frame_End_Sequence</td>
<td>2 Bit Times</td>
</tr>
<tr>
<td>Mini_Slot_Bits</td>
<td>3</td>
</tr>
</tbody>
</table>

VI. RESULTS FROM THE SIMULATION

The results obtained from the VisualSim tool are shown below. Fig. 4 shows a Time-Data plot of the channel activity contributed by 16 nodes. The plot has used different color coding for all 16 nodes. The latency plot shown in Fig.4
analyzes the delay with respect to the slot size i.e. the number of bytes per slot. It can be seen that as the bytes per slot increases, the latency also increases which means that for smaller slot size, the delay is less, faster is the response. Fig. 5 shows the bandwidth utilization and from the analysis of bandwidth utilization for channel 1 and channel 2 with respect to the bytes per slot (slot size), it has been proved that larger slot sizes increase the channel utilization.

VII. CONCLUSION AND FUTURE SCOPE

This paper initially detailed about the time-triggered architecture and its scope in the automotive domain, especially for the safety-critical applications. The paper then detailed about a specific protocol FlexRay, which has emerged as a guaranteed protocol for reliability in the vehicular industry. The main focus of this paper is on an effective analysis of the slot performance of the static segment of the FlexRay Protocol with VisualSim and the results confirm that as the bytes per slot increases, the latency also increases, i.e., for smaller slot size, the delay is less, faster is the response. From the bandwidth utilization analysis, it has been proved that as the slot sizes increase, a growth in the channel utilization can be observed.

As a future work, an investigation can be made on analyzing how optimum bandwidth utilization could be obtained without compromising the latency and thereby increasing the efficiency of the protocol.

REFERENCES


